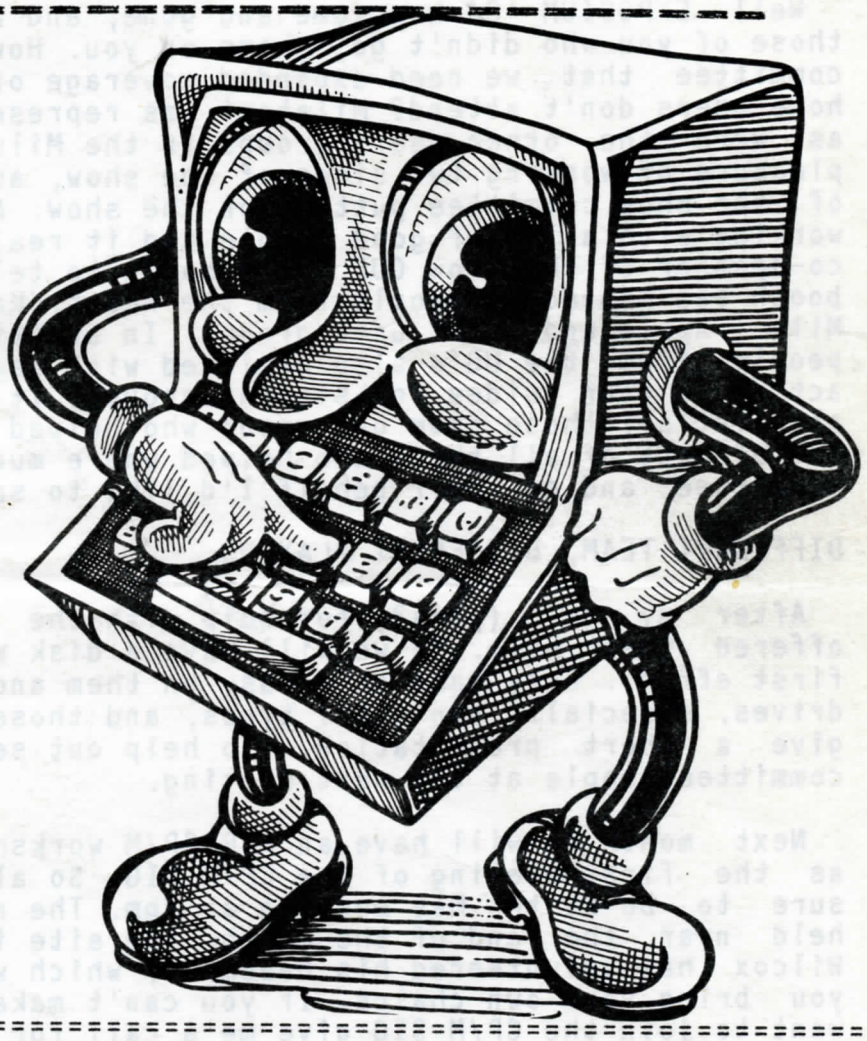
MILATRARI NEWSLETTER
VOLUME 3 NUMBER 6
MAY 1984
PRICE $1.00

MILWAUKEE AREA ATARI USERS CLUB



May meeting agenda
Saturday, May 19th

2:00 PM ATR8000 - CP/M SIG meeting

3:30 PM Business meeting

Annual Elections

Slate: President - Gary Nolan

Vice President - Chris Stieber

Secretary - Linda Scott

Treasurer - Steve Tupper

4:30 PM Demonstrations

Beginning on page 6 of this issue is a very special programming article by our own Erik Hanson. Erik is a senior at Menomonee Falls East High School. This programming project was entered in a competition with high school students through out the state. Erik earned a first place with this project. When you see Erik at the MILATARI meeting give him your congratulations. (Erik is also instructing the Assembly Language Class sponsored by MILATARI).

The ATR800 - CP/M Special Interest Group will meet at Don Wilcox's home on Wednesday, May 30th. The meeting will begin at 7:30PM. This months topic will be on the CP/M command structure. Don's home is located at 1419 W. Custer Ave., Milwaukee. Don asks that you bring your own folding chair.

PRESIDENT'S RAM

by Gary Nolan

IT TAKES A LOT OF PLAYERS TO MAKE A TEAM

Well EXPOSIUM '84 has come and gone, and a great show it was. For those of you who didn't go, shame on you. How can we convince the show committee that we need expanded coverage of the home market if the home users don't attend? Milatari was represented at the show by CUF, as were the other users groups in the Milwaukee area. I had the pleasure of working two sides of the show, as an exhibitor and as part of the show committee putting on the show. And I had the pleasure of working with a lot of good people and it really was a team effort. As co-sponsor of the show CUF had a booth to tell people who we are. That booth was manned by people from the Heath, Kaypro, Tandy, Osborne, TI, Milw. Apple and Atari user groups. In addition to manning that booth people from the UG's also assisted with registration, seminars and acted as tour guides for student groups. It would be a shame to try and list all those from our group who helped and miss even one person. The efforts of all those who helped where much appreciated by the show committee, and on their behalf I'd like to say a big THANK YOU!!!

DIFFERENT TEAM, DIFFERENT PLAYERS

After my call (plea?) for help with the workshops, three people offered their help. So we will have a disk workshop in June as their first effort. Lets make this easy on them and all you people with disk drives, especially non-Atari types, and those who work with disk files give a short presentation. To help out see one of the workshop committee people at the next meeting.

Next month we will have an ATR-CP/M workshop. This will also serve as the first meeting of the CP/M SIG. So all you ATR8000 people be sure to be at the May meeting at 2pm. The next SIG meeting will be held near the end of the month, the site is being worked on. Don Wilcox has volunteered his basement, which will hold 40 or more, if you bring your own chairs. If you can't make this months meeting but want to join the CP/M SIG give me a call for site info. You don't have to own an ATR to join, a CP/M computer or just an interest will do.

WANNA BUY A DUCK???

Those of you who expressed an interest in the Graph-fix labels for the fronts of the keys on the 800, take heart. We have ordered some and hope to have them by the 19th. No labels or ducks, eh!

(Continued on page 3)

PRESIDENT'S RAM (continued)

How about something called the MMS Atari Networking Switch. What this does is let you hook up to four Atari computers to one disk drive or printer. The switch works automatically without operator intervention. If purchased before Sept. 1st the price is $99, after that it goes to $125. It comes from Micro Systems Support in Lake Oswego, Oregon. See the info sheets at the meeting.

HOW MUCH???

Last month we received a bill for the use of Ambruster school that totaled (get ready) $150. That's right One Hundred Fifty Dollars! The time has come to think about breaking this group into two or three smaller groups. A Waukesha group would be a natural since there are a large number of members there. The new groups could be a little more responsive because of the smaller meeting sizes, but still have the advantages of a large base of membership to draw from. Think about it and if you would like to help get one of these groups going let me know.

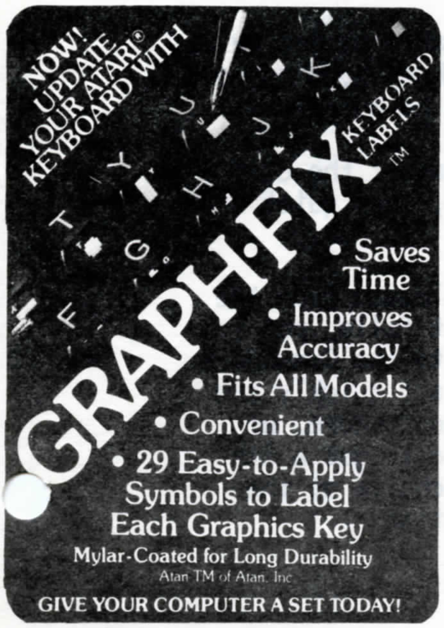
GET OUT AND VOTE!!

Remember this month is election month. So come to the meeting and VOTE.

SEE YOU ON THE 19th.........

= =

Sample of 'GRAFH*FIX keyboard labels to be sold by MILATARI to our membership for $3.25 per set.

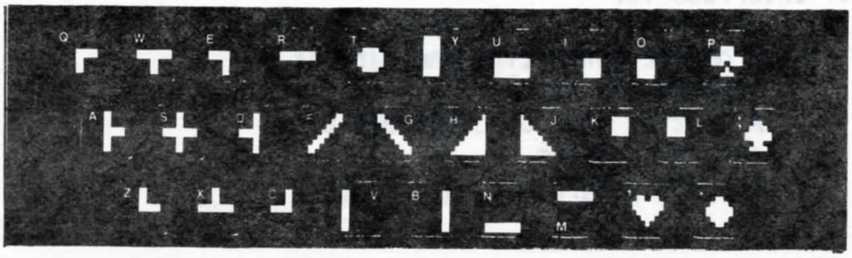


GRAPH·FIX KEYBOARD LABELS ™

These 29 high-quality, Mylar-coated symbols are easily applied to the front face of each graphics key and are sized to fit all models — both the 400 and 800 as well as the entire XL line. They will also work well with 400 add-on keyboards. These keyboard labels will aid you in programming by making the Control Key Graphics readily available without the need to consult various reference materials. GRAPH·FIX™ should help you save time, improve your accuracy, and provide a convenient way to include graphics symbols with your text. They are very durable and should last the life of your computer!

GIVE YOUR COMPUTER A SET TODAY – AND SHOW A FRIEND!

Graph-fix™ DOVESTAR Creative Concepts 1-409-727-5978



(ACTUAL SET)

Dovestar Creative Concepts, P. O. Box 2109, Nederland, Texas 77627

Test Yourself

How do you rate in the microcomputer world?

1. BASIC is:

   a. A stripped-down microcomputer model.
   b. The latest personal computer model from Radio Shack.
   c. Beginner's all-purpose symbolic instruction code.

2. Batch processing is:

   a. Placing all microcomputers in one centralized area.
   b. Collecting a group of programs and processing then one after the other.
   c. Group on-line access to a computer program.

3. A bit is:

   a. The smallest unit of information recognized by a computer.
   b. 10 bytes.
   c. A warning symbol for computer malfunctions.

4. A cursor is:

   a. An expression used by data processors when their system crashes.
   b. A hyphen-like or block indicator that shows where the next character will appear on a computer screen.
   c. A container for floppy disks.

5. Interface is:

   a. A device connecting one part of a computer to another.
   b. A meeting scheduled between two data processors.
   c. A human programming a computer.

6. Throughput is:

   a. A measure of the amount of work produced by a computer.
   b. Feeding paper through a computer printer.
   c. Loading new programs into a computer.

7. A loop is:

   a. Learning operator.
   b. Repetiton of a group of instructions in a computer program.
   c. Connection a mini- and microcomputer.

8. Mainframe is:

   a. A large computer.
   b. The plastic frame surrounding a computer.
   c. 100 microcomputers linked together.

Score 8-7: Have no fear you can talk the talk of computers.

6-5: You will be able to get by as long as you keep nodding your head and ask no questions.

4-3: Try to get out of the club. If you can't, keep quiet and smile.

2-0: Beg to be released from the club. If that fails, put in for a trade to the Commadore club. If we act quickly and don't show your score, we might be able to get a 6th round draft choice at the next CUF meeting.

1c, 2b, 3a, 4b, 5a, 6a, 7b, 8a

HARDWARE NOTES!

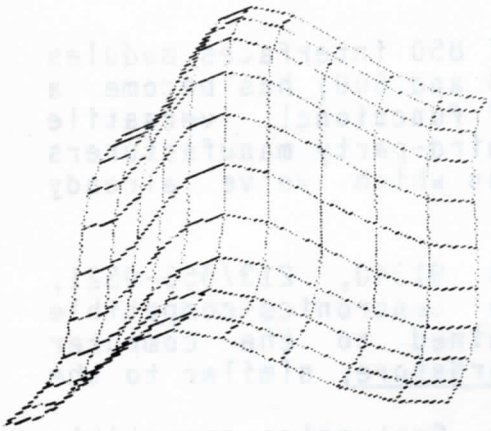
What last year was a severe shortage of ATARI 850 interfaces modules (used to connect printers and modems to the 400 and 800) has become a virtual famine. At the same time, the number of functional, versatile alternatives to the 850 continues to grow as third-party manufacturers rush to fill to vacuum. To the list of those on which we've already reported, we can add the following:

- AXIOM, 1014 Griswold Ave., San Fernando, CA 91340, 213/365-9521, offers two devices: the T-846, a simple Centronics-compatible printer interface which can be daisy-chained to the computer through a disk-drive ($99.95); and the AT-Wordstore, similar to the AT-846 but with a 32K buffer ($299).
- INTERFAST I is a buffered (4K), programmable Centronics-compatible printer interface from Advanced Interface Devices, P.O. Box 2188, Melbourne, FL 32902, 305/676-1275 (169.95).
- 1850 EXPANSION UNIT provides communication with Centronics-compatible printers and RS232 serial devices (modems) at transmission rates from 300 to 19,200 baud. Convologic, 421 Bay Tree Lane, Longwood, FL 32779, 305/869-6630. ($139).
- APE-FACE XLP. This stand-alone printer interface is daisy-chained through a disk drive, from Digital Devices, 151 6th St., Atlanta, GA 30313, 404/872-4430, ($89.95).
- A parallel interface through joystick ports 3 and 4 with disk-based transfer program and screen dump utility form Multivideo Services, Box 246, E. Elmhurst, NY 14051, 716/688-0469 ($49.95 plus $2 shipping).

BOOKS!

The first quarter of 1984 saw the appearance of several valuable Atari-related books:

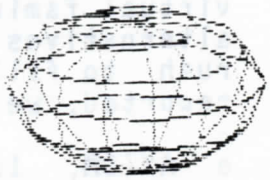
- PROGRAMS: Dilithium Press (8285 SW Nimbus, Beaverton, OR 97005, 800/ 547-1842) has published Tom Rugg, Phil Fredman and Timothy Barry's 32 BASIC PROGRAMS FOR THE ATARI COMPUTER (288 pages), covering a wide range of programming applications--games, education, mathematics and graphics. The book alone is $19.95; the book plus the programs on disk is $39.95.
- MORE PROGRAMS: One of the biggest bargins of the year has to be the A.N.A.L.O.G. COMPENDIUM, more than 50 programs (19 utilities, 21 graphics demonstrations, and 13 games) from the first ten issues of the magazine. For those who purchase the book, the programs are available on diskettew for an additional $35 (A.N.A.L.O.G., P.O. Box 23, Worcester, MA 01603).
- AND LOTS MORE INFORMATION: The ATARI USER'S ENCYCLOPEDIA by Gary Phillips and Jerry White (The Book Company, 11223 S. Hindry Ave., Los Angeles, CA 90445, 213/410-9466, 267 pages, $19.95) contains a monumental amount of valuable information on Atari computers and third party hardware and software. Devoted primarily to the 400 and 800 systems, the book also treats the XL series and the new DOS 3.0. It includes memory maps of the 400/800 and 1200XL computers and the addresses and telephone numbers of all major Atari-related manufacturers, publishers, vendors, and Atari user's groups.



3-D GRAPHICS

by

ERIK J. HANSON



program developed on
an ATARI 400 home computer
with
48K of memory
one ATARI 1050 disk drive
an ordinary television set
and one ALPHACOM 42 thermal printer

program written in
ATARI 8K BASIC COMPUTING LANGUAGE

ABSTRACT

This program, as the name 3-D GRAPHICS implies, is a program to plot 3-dimensional points and lines on a 2-dimensional computer screen. It can accept the coordinates of these points and lines in either cartesian or spherical coordinates, or it can generate the points and lines from a cartesian or spherical function.

3-D GRAPHICS allows the user to look at the system from any given point. The user can also vary the direction in which he is looking by either changing the direction of viewing or specifying a paticular point that he is looking towards. The user also specifies the desired angles for the field of vision. If a function is being used, the number of increments, the size of the increments, and the initial values for the two dependent variables must be selected by the user.

This program is limited by the facts that it is written for only one disk drive, and that 40K is the maximum amount of memory directly adressable with the ATARI 8K BASIC COMPUTING LANGUAGE. The program can be easily modified to take advantage of more disk drives, but the memory limitation can't be changed without changing the language that the program is written in. With one disk drive, the maximum numbers of points and lines for a system are 2430 and 1243 respectively. If a function is being used, the maximum number of points that it can generate is 4916. The program is designed to accomodate a large number of points and lines, so it does take a longer time than necessary when handling small systems.

Although there are some other programs like this on the market, all of the ones that I have seen have been limited compared to this one. I have seen few programs that allow functions, although changing the station point is somewhat common, I haven't seen such things as a direction of viewing and angles for a field of view in this type of a program. It can also handle larger systems than most other programs, and has some very nice editing features.

PROGRAM DESCRIPTION

In order to project these points, I had to build many calculations into this program. I decided to store all coordinates in cartesian form so that I could save the space that would be required if I used flags to indicate what (cartesian or spherical) each coordinate was stored in. I decided to have the option of using a function to generate a system, so that one would be able to save time typing and could easily examine some complicated functional relationships. I also had to figure out the neccessary calculations for displaying a system with all of the parameters that I wanted to use.

I started making a series of proofs. I started with a proof where the station point was at a point on the negative x axis looking towards the origon with no regard for a field of view. After many attempts and revisions, I made the first proof shown in appendix D. I then started working on other proofs, for two- and three- dimensional station points, using both rectangular and polar coordinates, all looking towards the origon. These proofs got very complicated, so I decided to try other methods to accomplish my goal. The first system was the one that was first created using the program. The second system is the same as the first system, except that the coordinate axes are moved so that the station point is at an arbitrary coordinate (XS) on the x axis looking towards the origon. To help better visualize the system, I chose XS to have a negative value in the program, but it could be either positive or negative. The fact that XS was arbitrary helped me with my angles for the field of view. Since my proofs give coordinates on the plane that contains the origon and is perpendicular to the line of sight, in other words the yz plane in the second system, I could pick a value for XS that would allow the actual screen width on this plane to be exactly within my horizontal field of view. I then used my vertical angle for my field of view to generate a scale for plotting y values. Because the computer plots little rectangles instead of little squares, there is another scale used in plotting the points and lines so that they have the correct ratio of height to width.

One of my biggest problems in the design of this program was deciding on how to store my variables. At first I used arrays for the coordinates, but quickly found myself running out of memory. Then I started rounding numbers to use three bytes by POKEing them into memory, but I still ran out of room. I was forced to use a disk drive to store my variables, so my program requires that the user have a disk drive.

I thought that this program was a challenge for me. I tried to use many programing REMarks in the program but, when I ran out of memory, I had to remove them. I tried to make a well structured program, and I feel satisified with 3-D GRAPHICS in this respect. I had to write many math proofs, and they were of the type that I had to do on my own, since it took too much time for others to understand what I was trying to do. I tried to make parts of the program, like the editing section, very menu driven. I would have liked to draw some of the lines that go off of the screen that 3-D GRAPHICS presently ignores, but I was unable to completely figure out the required proof, so the lines are in my list with the hope that at some later time someone can figure the problem out and then change the program, but they are not used when running the program. The only other major complaint that I have with my program is the fact that I ran out of memory before I had finished, and do not have an extensive error handler. I would have like to have it examine what type of error was encountered at what line, and then print an appropriate message, reset any messed up variables, and reenter the program at the correct line. I plan to revise this program as my ability increases, and especially after a course in vector analysis. In the future, with more powerful computers having more memory, I feel that this program can be used as a basis for a more extensive program, perhaps involving interrelations between three dimensional bodies in three dimensional space involving more functions, but right now it is not possible for me to write such a program on a home computer.

My program is a tool to allow a computer to project three dimensional points and lines onto a two dimensional computer screen. The points and lines can either be typed in by the user or generated by a function. Coordinates used can be either cartesian or spherical. If points and lines are typed in, the maximum numbers of each are 2430 and 1243 repectively. If a function is used, it can generate a maximum of 4916 points that can either be plotted or connected in three different ways. A function is also limited because it can must be less than 120 characters in length. Angles of vision for the field of view must, of course, be between 0 and 180 degrees.

The language that I wrote my program in is ATARI 8K BASIC COMPUTING LANGUAGE. My main reason for choosing this language is that it is the most common language on the market for the ATARI home computers. I also chose it so that it would be easier for others to read my program. Other reasons for choosing BASIC are its floating point package, trig functions, ability to use graphics directly, and I/O handling. I also prefer the language because I am very familiar with many of the shortcuts with it, like reading its six byte representation of numbers from the variable table.

The table below lists the time to convert from one system to the other and the time to plot the coordinates on the screen. The times given are in seconds, and the times for functions can be considered the same for converting systems as for points, but the plotting depends on the method for connecting the points.

TYPE OF SYSTEM	TIME TO CONVERT	TIME TO PLOT
POINTS		
1	8	2
10	16	4
100	96	36
1000	907	355
LINES		
1	9	2
10	26	6
100	2357	45

The following is a list of the main subroutines and subprogram used in my program. They are listed by their a name in capital letters that I will use for any future references in this text and a very brief description of what it does.

LOGO:Draws the title screem for the program
INIT:Initialization - sets up variables and files in program
END:Ending - closes files for quiting the program
ERROR:Error handler - tells type of error and where it happened
MENU:The main menu for the program
ROUTINES:The routines most often used by the subprograms
    ROUND:Rounds the variables X,Y,Z,A,B,C,D,E,F to two decimal places
    GETPOINT:Gets the coordinates of point I and stores in X,Y,Z
    GETLINE:Gets the start (A,B,C) and end (D,E,F) of line I
    PUTPOINT:Stores point I (X,Y,Z)
    PUTLINE:Stores line I (A,B,C) to (D,E,F)
    PUTDETAILS:Saves important variables for the system
    GETDETAILS:Gets important variables for the system (not used)
    ATN:Calculates the arc tangent of Y/X from 0 to 360 degrees
    GETFN:Gets the coordinates of functional point I (X,Y,Z)
    PUTFN:Saves the coordinates of functional point I (X,Y,Z)
    PUTFN:Saves the coordinates of functional point I (X,Y,Z)
    CLOSEFN:Closes functional disk file
    CS1:Changes from the origional to the altered system
    CS0:Changes from the altered to the origional system
    KEY:Waits until the user presses a key
    VIEW:Inputs the horizontal and vertical angles of vision
    DIRECTION:Inputs the angles for the direction of view or the point
        that the user is looking towards
    POLAR:Changes cartesian coordinates (X,Y,Z) to spherical (A,B,C)
    RECTANGULAR:Changes spherical (A,B,C) to cartesian (X,Y,Z)
    INPUT:Inputs the coordinates (X,Y,Z) for A$, converting to cartesian
        if neccessary
CREATESYSTEM:Subprogram to create a system of points and lines typed in
        by the user
DISPLAYSYSTEM:Subprogram to display a system
    DUMP:Routine to dump the screen to the printer
    DISPLAYNONFUNCTION:Routine to display a non-functional system
    PLOT:Routine to plot points (X,Y,Z)
    DRAW:Routine to draw lines (A,B,C) to (D,E,F)
    DISPLAYFUNCTION:Routine to display a function
        FTO:Routine to plot functional points

FT1:Routine to plot points, connecting same functional x
FT2:Routine to plot points, connecting same functional y
FT3:Routine to plot points, connecting in a grid (FT1&FT2)
ADDTOSYSTEM:Subprogram to add to a user created system
ADDPOINTS:Routine to allow the user to add points
ADDLINES:Routine to allow the user to add lines
ADDSUBMENU:The menu for this subprogram
EDITSYSTEM:Subprogram that allows the user to view and/or change the the program parameters, and view, change, and/or delete user-created point and/or lines
VIEW:Routine to view parameters and/or coordinates
CHANGE:Routine to change parameters and/or coordinates
DELETE:Routine to delete parameters and/or coordinates
SUBMENU1:The first menu for this subprogram
SUBMENU2:The second menu for this subprogram
SUBMENU3:The third menu for this subprogram
FUNCTION:Subprogram to create a system from a function
FNDATA:All of the data needed for the system is input here
CALCULATE:Generates the points for the system from the function
FTMENU:Menu used to determine the method of plotting

On the following pages is what I consider to be a detailed block diagram of my program. It is not a flowchart, and I do not have any flowcharts in the appendicies. It is meant to be used as a guide to understand how my program works in a general sense.

INPUT/OUTPUT

This section describes the manner in which the program handles input and output. It describes the different methods that the user must use to correctly interface with the program, and what happens if the user attempts a method other than the desired one. I have tried to limit the number of different input/output formats so that the user can feel more at ease with the program in a shorter amount of time.

The first form of input request that the user encounters, and the simplest, is the request to hit any key. Although this may sound self-explanitory, an user who is not familiar with the computer may not realize certain things. The program is actually asking the user to hit any alphanumeric key on the keyboard. Special function keys like OPTION, SELECT, and START will not do anything. Keys like CTRL that would not actually print a character will not work either. And the SYSTEM RESET and BREAK keys will stop the program from running. Even though this case will almost always be understood, some people who are not familiar with computers may have trouble, and it is therefore mentioned to help them with the program.

After the user hits a key, the program will go to its main menu. The user will then be asked to type the number of his/her chioce. After the user types the number, he/she must hit the RETURN key. Although this is not stated in the program, it is assumed that the user is familiar with computers to the extent that he/she will know enough to hit the RETURN key. If the user types in a numerical choice other than those listed, the program will redraw the menu and ask for another choice. If the user just hits the RETURN key, or enters a non-numerical input, the program will display an error message, wait for a keypress, and then return to the main menu. From this point on, anyplace in the program where a numerical input is asked for, the method of entering the choice will be basically the same as it is for the main menu. The only real difference is that with numbers like coordinates that do not have to be within a certain range, any numerical value will be accepted.

If the user chooses a subprogram that is menu driven, he/she will be asked to choose a letter indicating a choice. All that he/she has to do is to hit the key that stands for the correct choice. If a key (one that would work for HIT ANY KEY) that does not stand for one of the menu options is hit, the menu will be redrawn, and the user will be asked to choose an option from the menu. This same method is used with all of the submenus in the program. In parts of the program the user may be asked to type Y to perform a function. Typing Y will perform the function, and typing any other key will abort the operation. Again the keys are those allowed for the HIT ANY KEY choice, and RETURN is not needed. Since capital letters are shown in the choices, and are required to run the program, they are the only expected form for the

letter inputs.

In part of the program, the user is asked to type in a function for Z. The user is the expected to type in the command as if it were a BASIC statement. The only editing key that the user can use is the BACK S key, because the control keys will affect the input. The statement is typed in and the RETURN key is hit when the statement is finished. If the statement does not make sense to BASIC, an error message will show up, the user will be requested to hit a key, and the program will be back at the main menu.
Since this program is presently a more mathematical than physical program, units are not too important. All of the angle measures are given in degrees because it is easier for most people to picture things in degrees than it is for them to use radians. Since the coordinates are relative to one another, the units used for all coordinates must be the same in order to get an actual representation. These are the only rules for units in a system typed in by the user. If a function is used, however, care must be taken so that the units of x and y agree with the units of any constants in the function. If this is not done, it could result in an incorrect functional graph, and the problem may take quite a while to figure out.

There are two different files that are always open in this program. They are not neccessary, but they do allow that program to be expanded upon very easily. The first file (#1) is used to recieve input directly from the keyboard. By changing one command, the input for certain parts of the program, like the submenus, could come from another device, like a modem. The second file (#2) is open to the screen editor. All printing goes to it, and it interpets the data and prints it on the screen. This could be changed to allow printing to the printer, but parts of the statements would have to be changed, and the input would not presently be shown. Nevertheless, it could prove very helpful to someone adding to the program later on, so I chose to keep it in.

There are also three disk files that are sometimes open (#3) in my program. The first one, D:DETAILS, is used to save the parameters for the system. although it is possible to retrieve these parameters, there is no place in the program where this is done. There is a routine to do this in the program, however, so that these parameters could be retrieved from a work disk. At this time, D:DETAILS is not neccessary, but it is included so that future expansion can be easier. The other files, D:POINTS and D:LINES, are neccessary. For a system typed in by the user, these files are self-explanitory. For a system created from a function, D:POINTS is used to store the first points and, if more room is needed, D:LINES is used to store the rest of the points (2 points/line).

All three disk files put together take up a whole disk. While D:DETAILS is only one sector (128bytes/sector-124useable bytes/sector), D:POINTS is 350 sectors and D:LINES is 356 sectors. In all of the files floating point numbers are stored as a 6-byte representation. D:POINTS stores the points in order, listing each point as x, y, and z coordinates. D:LINES stores the lines in order, listing each line by a starting and ending point. D:DETAILS stores important numbers. First it uses one byte each to store CS, FN, and FT. Then it uses two bytes each to store NP and NL. Then it uses the 6-byte representation to store:the coordinates of the station point, the direction of view, the coordinates of the point that you're looking towards, the angles of vision for the field of view, the x offset for the station point, the initial values, sizes of increments, and numbers of increments for the functional x and y respectively.

Throughout this program, I have tried to keep the methods of input the same in each of the subprograms. My main reason for doing this is so that the user will have an easier time becomming comfortable with my program. I feel that if the user is more comfortable with the program, the user will also be more productive with it. I believe that this program, as written, is a very prowerful tool, but I also believe that it can be made even more powerful for some users by tailoring it to a specific application. It is for this reason that I have tried to structure it so that expanson is easy, but whoever is to be expanding it should preferably try to keep the same structure throughout, otherwise one of the best aspects of the program will be lost.

SAMPLE OUTPUT

On the last two pages of this section, there are photo-copies of the screen dumps of actual images created by this program. In this part of the documentation, I will descripe what type of a system was used to create each projection. I chose these examples because they show most of the capabilities of this program. I could not, of course, present every possible option on two pages, but I feel that I have a fairly good representation of the capabilities of my program.

The first group of graphs (1A thru 1E) are all of the cartesin function z=3sin3θx + 6cos3θy. The function has the initial values of both x and y at -5. The sizes of the increments for both x and y are 1, and the numbers of both x and y increments are 10 each. This results in a 121 point square (in the x,y plane) graph of the function. The function is to be plotted by connecting the points in a x,y grid.

The graph labled 1A is the projection of this system as seen from a station point at the cartesian point (-30,15,50) looking towards the

cartesian point at (1,-1,1) with 40 degree fields of view both horizontally and vertically. The graph labled 1B is the save graph as 1A, but the fields of view are each only 30 degrees, so the projection appears larger because less area has to fit onto the same size screen. The next graph in this set, 1C, is the same as 1B except for the fact that the station point has moved from the cartesian point (-30,15,50) to the cartesian point (-30,15,100). The graph 1D appears to be the graph 1C, but a close look shows that it is actually shifted slightly. It is the same as 1C, but instead of looking towards the cartesian point (1,-1,1), you are looking towards the cartesian point (-3,0,0). The last graph, 1E, is the same as 1D, but the graph is streched vertically. This is done by using two different angles for the field of view. While the horizontal angle is still at 30 degrees, the vertical angle has been decreased to 10 degrees. This ability is very useful for examining functional relationships.

The second group of graphs (2A thru 2C) are all of the cartesian function z=x*x+y*y. The function, like the last one, has the initial value of x an the initial value of y both equal to -5. The sizes of both the x and y increments are 2, and the numbers of increments are five each. The resulting system is a 36 point sqruare on the x,y plane. The function is to be plotted by connectinng the same x coordinates.

The graph labled 2A is the projection of this system as seen from a station point at the cartesian point (-30,15,20) looking towards the cartesian point (-1,1,-1) with a horizontal angle of vision of 120 and a vertical angle of vision of 160 degrees. The graph labled 2B is the same as 2A except for the facts that you are now looking towards the cartesian point at (-1,1,25) and the angles for the field of view have been changed to 45 degrees horizontally and 120 degrees vertically. The graph of 2C is the same graph as 2B, but instead of just connecting the same x coordinate, the same x and the same y coordinates are connected in a grid.

The third group of graphs (3A thru 3D) are all of the same system that was typed in by the user. This system consists of a point at the origon that is directly in the center of a 2X2X2 square. The square is made up of twelve lines that, using cartesian coordinates, have endpoints at the eight possible points given as (+/-1,+/-1,+/-1). Each one of these lines was typed in by the user along with the point in the center.

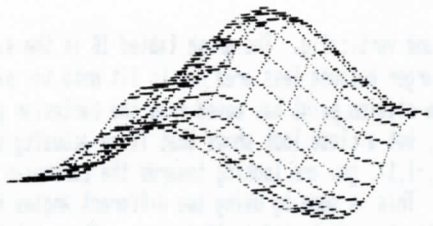
The graph 3A shows this system as seen from a cartesian station point at (-15,5,8) looking towards the origon with horizoltal and vertical angles for the field of view of 45 degrees each. In the second point, 3B, the horizontal angle for the field of view has been changed to 20 degrees to show the effect of streching that a change like this can produce. In 3C the angles for the field of view are both set at 30 degrees, and you are now looking towards the cartesian point (-1,1,-2). Notice how objects distort when you view them from an angle. This was my main motivation for writing this program, I love the fact that I can now observe and/or measure this incredible effect. The graph 3D is the same as this graph with the exception that the station point is now at the cartesian point (-25,0,8).

The last set of graphs (4A thru 4F) are of a very simple polar function. The function is r=4, so it produces a ball. The xy angle (X in the program or theta in mathematics) starts at 0 degrees and a total of twelve increments of 30 degrees each are added to end with a 360 degree angle. The (xy)z angle (Y in the program or phi in mathematics) starts at -90 degrees and a total of six increments of 30 degrees each are added to end with a 90 degree angle. I chose to connect the points that had the same functional y (phi) value.

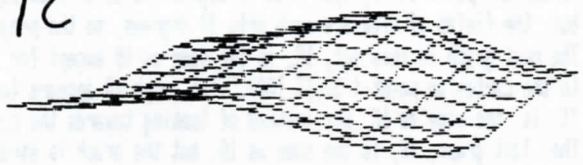
In the graph 4A, the system is seen from a cartesian station point at (-15,5,8) looking towards the cartesian point at (-1,1,-1) with 60 degree angles of view both vertically and horizontally. The graph 4B is the same as 4A, except that the station point is moved to the cartesian point (-10,10,10). The graph 4C is the same 4A except that the angles for the field of view are both set to 75 degrees. 4D is the same graph as 4C, but the functional coordinates are now connected in a grid. The same thing is done in 4E, which is the same as 4A except for the plotting method. 4F is the same graph as 4E, except for the fact that you are looking towards the cartesian point (0,0,0) instead of the cartesian point (-1,1,-1).

As I have already said, this is just a sample of the capabilities of the program. There are actually much larger systems, more complicated functions, and many different ways of designating parameters that those shown here. This section is meant to give examples of some of the easier uses so that the user has a better idea of what to expect and of how to use the program to his/her best advantage.

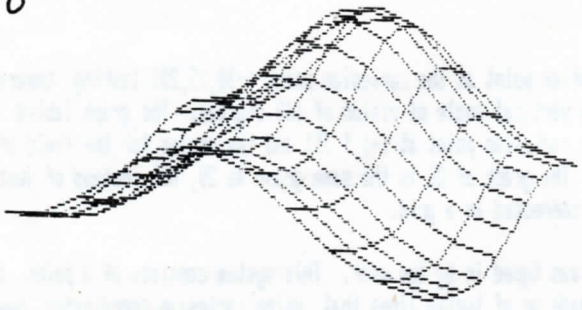
1A



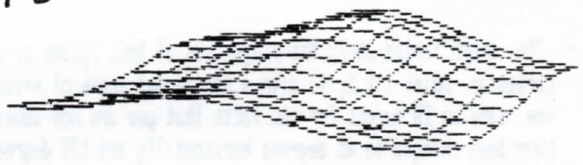
1C



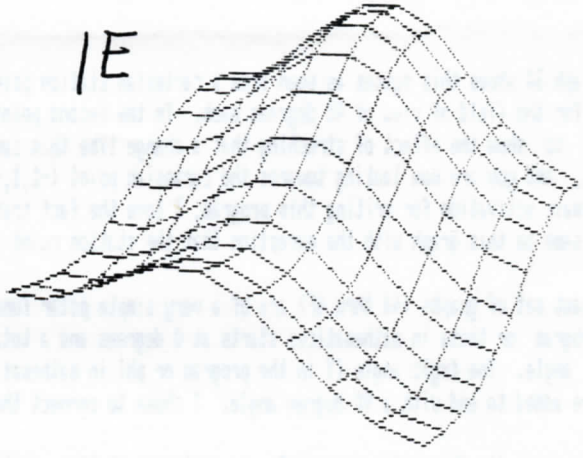
1B



1D



1E

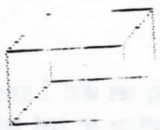


2A

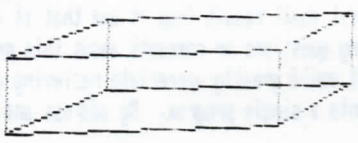
2B

2C

3A



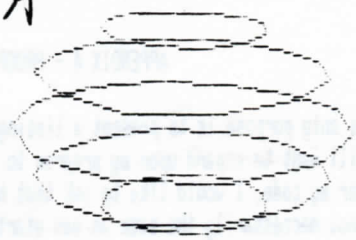
3B



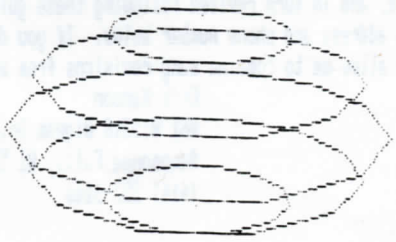
3C

3D

4A



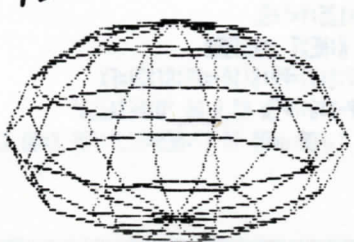
4B



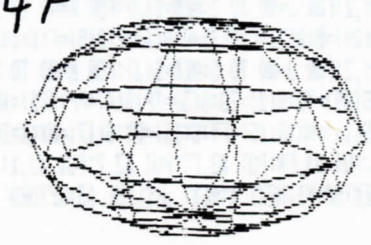
4C

4D

4E



4F



APPENDIX A - PROGRAM LISTING

This sections only purpose is to present a listing of my program so that others may see what I have done and how I have done it. Hopefully some users will want to expand upon my program to make it better for a specific purpose or just better in every respect. If the user does decide to alter my code, I would like to ask that however it is altered, please take care to let it use the same input structure throughout the program (not neccessarily the same as was started with) so that it is easier for someone to get used to the program.

It should be noted that this program and this documentation each carry copyrights. Any action involving this program or this documentation in which a party knowingly uses any part or result of this program or the documentation to secure a private profit is hereby prohibited. Duplication and distribution of this program or the documentation to others on a non-profit basis is allowed. Any expansion or revision of this program or documentation is allowed provided that no intentional profit shall result from it and that it is free to be duplicated, distributed, and in turn revised following these guidelines. If you have any questions or comments about this program feel free to contact me at the address and phone number below. If you do revise this program, I would greatly appreciate recieving a copy of your revision as this would allow me to combine many revisions from many different sources into a single program. My address and phone number are:

Erik Hanson
N61 W15318 Wigwam Dr.
Menomonee Falls, WI 53051
(414) 252-3146

```
1 REM 3-D GRAPHICS
2 REM COPYRIGHT 1984
3 REM BY ERIK HANSON
4 REM ALL RIGHTS RESERVED
10 GOTO 200
100 GRAPHICS 18:POSITION 4,3:? #6;"3-D GRAPHICS":POSITION 3,5:? #6;"by erik hanson":POSITION 3,7:? #6;"COPYRIGHT 1984":RETURN
200 CLR :MEM=1:DIM A$(110),B$(18),C$(16),D$(13),P$(8),L$(7),S(17):SM=PEEK(134)+256*PEEK(135)+2:MP=2430:ML=1243
230 TRAP 400:DEG :OPEN #1,4,0,"K:":OPEN #2,8,0,"E:":GOSUB 100:POSITION 3,10:? #6;"FLIP DISK OVER":SP=369:SL=4:Q=256:Q0=0:Q3=0.5:Q4=100
245 Q9=1:CX=159:CY=148:GC=0.64706:EP=718:EL=359:EN=123:PI=3.14159265:B$="the start of line ":C$="the end of line "
252 D$="point number ":P$="D:POINTS":L$="D:LINES":POSITION 3,11:? #6;"Hit any key      ":GET #1,A:GOSUB 500:GOSUB 300:END
300 CLOSE #1:CLOSE #2:GOSUB Q4:FOR I=Q0 TO Q:NEXT I:RETURN
400 IF PEEK(195)>127 THEN CLOSE #3
405 POKE 559,34:TRAP 400:? #2;"There was an error-";PEEK(195)
410 ? #2;"at line ";PEEK(186)+Q*PEEK(187);"!"
420 GOSUB 900
500 GRAPHICS Q0:SETCOLOR 2,Q0,Q0:? #2;"1=Create a new system":? #2;"2=Display system":? #2;"3=Add to system"
517 ? #2;"4=Edit system":? #2;"5=Create system from function":? #2;"6=Quit"
521 ? #2;"Your choice";:INPUT A:IF A<Q9 OR A>6 OR A>INT(A) THEN 500
523 IF A=6 THEN RETURN
524 ? #2;" ":GOSUB A*1000+(A=4)*1000+(A=5)*3000:GOTO 500
700 X=INT(X*Q4+Q3)/Q4:Y=INT(Y*Q4+Q3)/Q4:Z=INT(Z*Q4+Q3)/Q4:A=INT(A*Q4+Q3)/Q4:B=INT(B*Q4+Q3)/Q4
701 C=INT(C*Q4+Q3)/Q4:D=INT(D*Q4+Q3)/Q4:E=INT(E*Q4+Q3)/Q4:F=INT(F*Q4+Q3)/Q4:RETURN
710 X=I*18-18:Y=INT(X/125):Z=X-Y*125:Y=Y+SP
711 POINT #3,Y,Z:FOR J=Q0 TO 2:FOR K=Q0 TO 5:GET #3,X:POKE SM+K,X:NEXT K:S(J)=MEM:NEXT J:X=S(Q0):Y=S(Q9):Z=S(2):RETURN
720 X=I*36-36:Y=INT(X/125):Z=X-Y*125:Y=Y+SL:POINT #3,Y,Z
721 FOR J=Q0 TO 5:FOR K=Q0 TO 5:GET #3,X:POKE SM+K,X:NEXT K:S(J)=MEM:NEXT J:A=S(Q0):B=S(Q9):C=S(2):D=S(3):E=S(4):F=S(5):RETURN
730 S(Q0)=X:S(Q9)=Y:S(2)=Z:X=I*18-18:Y=INT(X/125):Z=X-Y*125:Y=Y+SP
731 POINT #3,Y,Z:FOR J=Q0 TO 2:MEM=S(J):FOR K=Q0 TO 5:PUT #3,PEEK(SM+K):NEXT K:NEXT J:RETURN
740 S(Q0)=A:S(Q9)=B:S(2)=C:S(3)=D:S(4)=E:S(5)=F:X=I*36-36:Y=INT(X/125):Z=X-Y*125:Y=Y+SL
741 POINT #3,Y,Z:FOR J=Q0 TO 5:MEM=S(J):FOR K=Q0 TO 5:PUT #3,PEEK(SM+K):NEXT K:NEXT J:RETURN
750 S(Q0)=XO:S(Q9)=YO:S(2)=ZO:S(3)=TP:S(4)=PP:S(5)=XP:S(6)=YP:S(7)=ZP:S(8)=TV:S(9)=PV:S(10)=XS:S(12)=FX
751 S(13)=DX:S(14)=NX:S(15)=FY:S(16)=DY:S(17)=NY:X=INT(NP/Q):Y=INT(NL/Q):Z=NP-X*Q:OPEN #3,8,Q0,"D:DETAILS"
752 PUT #3,CS:PUT #3,FN:PUT #3,FT:PUT #3,Z:PUT #3,X:Z=NL-Y*Q:PUT #3,Z:PUT #3,Y:FOR J=Q0 TO 17:MEM=S(J):FOR K=Q0 TO 5
753 PUT #3,PEEK(SM+K):NEXT K:NEXT J:CLOSE #3:RETURN
```

```
755 OPEN #3,4,0,"D:DETAILS":GET #3,CS:GET #3,FN:GET #3,FT:GET #3,X:GET #3,Y:NP=X+0*Y:GET #3,X:GET #3,Y:NL=X+0*Y:FOR J=0 TO 17
756 FOR K=0 TO 5:GET #3,X:POKE SM+K,X:NEXT K:S(J)=MEM:NEXT J:CLOSE #3:XO=S(0):YO=S(9):ZO=S(2):TP=S(3):PP=S(4):XP=S(5):YP=S(6)
757 ZP=S(7):TV=S(8):PV=S(9):XS=S(10):FX=S(12):DX=S(13):NX=S(14):FY=S(15):DY=S(16):NY=S(17):RETURN
760 IF X=0 THEN Z=90+180*(Y<0):RETURN
761 Z=ATN(ABS(Y/X))
762 IF X<0 THEN 765
763 IF Y>=0 THEN RETURN
764 Z=360-Z:RETURN
765 IF Y>=0 THEN Z=180-Z:RETURN
766 Z=180+Z:RETURN
770 IF I>MP AND OI<=MP THEN POINT #3,EP,EN:CLOSE #3:OPEN #3,4,0,L$
771 IF I<=MP AND OI>MP THEN POINT #3,EL,EN:CLOSE #3:OPEN #3,4,0,P$
772 IF I>MP THEN X=(I-MP)*18-18:Y=INT(X/125):Z=X-Y*125:Y=Y+SL:GOTO 711
773 IF I<=MP THEN GOTO 710
780 IF I>MP AND OI<=MP THEN POINT #3,EP,EN:CLOSE #3:OPEN #3,12,0,L$
781 IF I<=MP AND OI>MP THEN POINT #3,EL,EN:CLOSE #3:OPEN #3,12,0,P$
782 IF I>MP THEN S(0)=X:S(9)=Y:S(2)=Z:X=(I-MP)*18-18:Y=INT(X/125):Z=X-Y*125:Y=Y+SL:GOTO 731
783 IF I<=MP THEN GOTO 730
790 IF I<=MP THEN POINT #3,EP,EN:CLOSE #3:RETURN
791 IF I>MP THEN POINT #3,EL,EN:CLOSE #3:RETURN
800 POKE 559,0:CS=9:GOSUB 750:IF FN=9 THEN 810
801 IF NP>0 THEN OPEN #3,12,0,P$:FOR I=9 TO NP:GOSUB 710:GOSUB 820:GOSUB 730:NEXT I:POINT #3,EP,EN:CLOSE #3
802 IF NL=0 THEN POKE 559,34:RETURN
803 OPEN #3,12,0,L$:FOR I=9 TO NL:GOSUB 720:X=A:Y=B:Z=C:GOSUB 820:A=X:B=Y:C=Z:X=D:Y=E:Z=F:GOSUB 820:D=X:E=Y:F=Z:GOSUB 740:NEXT I
804 POINT #3,EL,EN:CLOSE #3:POKE 559,34:RETURN
810 OPEN #3,12,0,P$:OI=0:FOR I=9 TO (NX+1)*(NY+1):GOSUB 770:GOSUB 820:GOSUB 780:OI=I:NEXT I:POKE 559,34:GOTO 790
820 X=X-XO:Y=Y-YO:Z=Z-ZO:GOSUB 950:B=B-TP:C=C-PP:GOSUB 960:X=X+XS:RETURN
850 POKE 559,0:CS=0:GOSUB 750:IF FN=9 THEN 860
851 IF NP>0 THEN OPEN #3,12,0,P$:FOR I=9 TO NP:GOSUB 710:GOSUB 870:GOSUB 730:NEXT I:POINT #3,EP,EN:CLOSE #3
852 IF NL=0 THEN POKE 559,34:RETURN
853 OPEN #3,12,0,L$:FOR I=9 TO NL:GOSUB 720:X=A:Y=B:Z=C:GOSUB 870:A=X:B=Y:C=Z:X=D:Y=E:Z=F:GOSUB 870:D=X:E=Y:F=Z:GOSUB 740:NEXT I
854 POINT #3,EL,EN:CLOSE #3:POKE 559,34:RETURN
860 OPEN #3,12,0,P$:OI=0:FOR I=9 TO (NX+9)*(NY+9):GOSUB 770:GOSUB 870:GOSUB 780:OI=I:NEXT I:POKE 559,34:GOTO 790
870 X=X-XS:GOSUB 950:B=B+TP:C=C+PP:GOSUB 960:X=X+XO:Y=Y+YO:Z=Z+ZO:RETURN
900 ? #2;" ":? #2;"Hit any key to continue":GET #1,A:RETURN
910 ? #2;"What is your horizontal (x,y) angle  of vision (0<angle<180)";:INPUT TV
911 ? #2;"What is your vertical (x,z) angle of  vision (0<angle<180)";:INPUT PV
912 A=SIN(TV/2)/COS(TV/2):B=SIN(PV/2)/COS(PV/2):SY=(CY-1)/(CX-1)*A/B:XS=-CX/A:RETURN
920 ? #2;"Type 1 if looking towards a point        2 if looking in a direction":? #2;"Your choice ";:INPUT P
921 IF P>INT(P) OR P<9 OR P>2 THEN 920
922 IF P=2 THEN P=0:GOTO 925
923 A$="the point":GOSUB 990:XP=X:YP=Y:ZP=Z:IF XO<>XP THEN X=X-XO:Y=Y-YO:Z=Z-ZO:GOSUB 950:TP=B:PP=C:RETURN
925 ? #2;"What are the angles (xy,xz) of the   direction that you are looking in";:INPUT TP,PP:RETURN
950 A=SQR(X*X+Y*Y+Z*Z):D3=Z:GOSUB 760:B=Z:D4=Y:Y=D3:D3=X:X=SQR(X*X+D4*D4):GOSUB 760:C=Z:Z=Y:Y=D4:X=D3:RETURN
960 X=A*COS(B)*COS(C):Y=A*SIN(B)*COS(C):Z=A*SIN(C):RETURN
990 ? #2;"Type 1 for rectangular coordinates    or 2 for polar coordinates (degrees)  for ";A$;
991 INPUT A:IF A<9 OR A>2 THEN 990
992 IF A=2 THEN 994
993 ? #2;"What are the coordinates (x,y,z) for  ";A$;:INPUT X,Y,Z:RETURN
994 ? #2;"What are the coordinates (r,theta,phi)for ";A$;:INPUT A,B,C:GOTO 960
1000 CS=0:FN=0:? #2;"}";:A$="the station point":GOSUB 990:XO=X:YO=Y:ZO=Z:GOSUB 920
1001 ? #2;"How many points, and how many lines  for this system";:INPUT NP,NL
1002 IF NP>INT(NP) OR NP<0 OR NL>INT(NL) OR NL<0 THEN 1001
1003 IF NP>MP OR NL>ML THEN ? #2;"Not enough room!":GOTO 1050
```

```
1004 IF NP=0 THEN 1006
1005 A$=D$:OPEN #3,12,0,P$:FOR I=0 TO NP:A$(14)=STR$(I):GOSUB 990:GOSUB 730:NEXT I:POINT #3,EP,EN:CLOSE #3
1006 IF NL=0 THEN 1009
1007 OPEN #3,12,0,L$:FOR I=0 TO NL:A$=B$:A$(19)=STR$(I):GOSUB 990:D1=X:D2=Y:D3=Z:A$=C$:A$(17)=STR$(I)
1008 GOSUB 990:A=D1:B=D2:C=D3:D=X:E=Y:F=Z:GOSUB 740:NEXT I:POINT #3,EL,EN:CLOSE #3
1009 GOSUB 910:FX=0:DX=FX:NX=DX:FY=NX:DY=FY:NY=DY:FT=NY:GOSUB 750:RETURN
2000 IF CS=0 THEN GOSUB 800
2001 GOSUB 2500:GRAPHICS 24:SETCOLOR 2,0,0:COLOR 9:IF FN=0 THEN GOSUB 2100
2055 IF FN=9 THEN GOSUB 2400
2060 POKE 53279,0
2063 IF PEEK(53279)<6 THEN RETURN
2065 IF PEEK(53279)>6 THEN 2063
2070 X=PEEK(560)+0*PEEK(561):X=PEEK(X+4)+0*PEEK(X+5):OPEN #3,0,68,"P:":FOR I=0 TO 7679:J=PEEK(X+I)
2075 PUT #3,J-(J=155):NEXT I:CLOSE #3:GOTO 2060
2100 IF NP=0 THEN 2115
2104 OPEN #3,4,0,P$:FOR I=1 TO NP:GOSUB 710:GOSUB 2210:NEXT I:POINT #3,EP,EN:CLOSE #3
2115 IF NL=0 THEN RETURN
2116 OPEN #3,4,0,L$:FOR I=1 TO NL:GOSUB 720:GOSUB 2220
2131 NEXT I:POINT #3,EL,EN:CLOSE #3:RETURN
2150 IF ABS(D)>ABS(XS) AND SGN(D)=SGN(XS) THEN RETURN
2151 N=0:GOSUB 2350:GOTO 2200
2200 IF N=1 THEN 2230
2201 RETURN
2210 IF ABS(X)>ABS(XS) AND SGN(X)=SGN(XS) THEN RETURN
2214 J=Y*XS/(XS-X):K=Z*XS/(XS-X):IF ABS(J)<CX AND ABS(K)<SY*CY THEN PLOT CX+J,(CY-SY*K)*GC
2216 RETURN
2220 IF ABS(A)>ABS(XS) AND SGN(A)=SGN(XS) THEN 2150
2222 D1=B*XS/(XS-A):D2=C*XS/(XS-A):IF ABS(D1)>CX OR ABS(D2)*SY>CY THEN 2200
2225 PLOT CX+D1,(CY-SY*D2)*GC
2230 IF ABS(D)>ABS(XS) AND SGN(D)=SGN(XS) THEN 2250
2232 D3=E*XS/(XS-D):D4=F*XS/(XS-D):IF ABS(D3)>CX OR ABS(D4)*SY>CY THEN 2250
2235 DRAWTO CX+D3,(CY-SY*D4)*GC:RETURN
2250 N=9:GOSUB 2350:RETURN
2350 J=(E-B)/(D-A):K=(F-C)/(D-A):X=(CX-B+J*A)/(J+CX/XS):GOSUB 2370:X=-(CX+B-J*A)/(J-CX/XS):GOSUB 2370
2351 X=(CY-C+K*A)/(K+CY/XS/SY):GOSUB 2370:X=-(CY+C-K*A)/(K-CY/XS/SY):GOTO 2370
2370 IF (X>A AND X>D) OR (X<A AND X<D) THEN RETURN
2371 Y=J*(X-A)+B:Z=K*(X-A)+C:N=N+1:G=Y*XS/(XS-X):H=Z*XS/(XS-X)
2373 IF ABS(G)>CX OR ABS(H)*SY>CY THEN RETURN
2377 IF N=1 THEN PLOT CX+G,(CY-H*SY)*GC
2378 IF N=2 THEN DRAWTO CX+G,(CY-H*SY)*GC
2379 RETURN
2400 GOTO 2410+FT*20
2410 OPEN #3,4,0,P$:I=0:FOR R=0 TO NX:FOR S=0 TO NY:OI=I:I=R*NY+R+S+9:GOSUB 770:GOSUB 2210:NEXT S:NEXT R:GOTO 790
2430 OPEN #3,4,0,P$:I=0:FOR R=0 TO NX:OI=I:I=R*NY+R+9:GOSUB 770:GOSUB 2210:D=X:E=Y:F=Z:FOR S=9 TO NY:OI=I:I=R*NY+R+S+9
2431 GOSUB 770:A=D:B=E:C=F:D=X:E=Y:F=Z:GOSUB 2220:NEXT S:NEXT R:GOTO 790
2450 OPEN #3,4,0,P$:I=0:FOR S=0 TO NY:OI=I:I=S+9:GOSUB 770:GOSUB 2210:D=X:E=Y:F=Z:FOR R=9 TO NX:OI=I:I=R*NY+R+S+9
2451 GOSUB 770:A=D:B=E:C=F:D=X:E=Y:F=Z:GOSUB 2220:NEXT R:NEXT S:GOTO 790
2470 GOSUB 2430:GOTO 2450
2500 ? #2;"}     This subprogram will display yourpresent system.  When it is finished"
2510 ? #2;"plotting, the keyboard speaker will":? #2;"sound a click.  You can then view the"
2520 ? #2;"picture for as long as you choose. If you enter the attract mode, just hit a";
2530 ? #2;"key to avoid it.  If you want to dump":? #2;"the screen to the printer, hit the"
2540 ? #2;"START key.  When you are finished withthe picture, hit either the OPTION or"
2550 ? #2;"the SELECT key to return to the main":? #2;"program menu.":GOTO 900
```

```
3000 IF FN=0 THEN ? #2;"This is a function, not a system":GOTO 900
3005 IF CS=0 THEN GOSUB 850
3020 GOSUB 3600:IF A=80 THEN 3100
3030 IF A=76 THEN 3300
3040 IF A=81 THEN RETURN
3050 GOTO 3020
3100 ? #2;"How many points do you want to add";:INPUT N:IF N>INT(N) OR N<0 THEN 3100
3115 IF N=0 THEN 900
3120 IF NP+N>MP THEN ? #2;"There is not enough room":GOTO 900
3125 A$=D$:OPEN #3,12,0,P$:FOR I=NP+0 TO NP+N:A$(17)=STR$(I):GOSUB 990:GOSUB 730:NEXT I:NP=NP+N:POINT #3,EP,EN:CLOSE #3:GOTO 900
3300 ? #2;"How many lines do you want to add";:INPUT N:IF N>INT(N) OR N<0 THEN 3300
3315 IF N=0 THEN 900
3320 IF NL+N>ML THEN ? #2;"There is not enough room":GOTO 900
3325 OPEN #3,12,0,L$:FOR I=NL+0 TO NL+N:A$=B$:A$(19)=STR$(I):GOSUB 990:D1=X:D2=Y:D3=Z:A$=C$:A$(17)=STR$(I)
3390 GOSUB 990:A=D1:B=D2:C=D3:D=X:E=Y:F=Z:GOSUB 740:NEXT I:POINT #3,EL,EN:CLOSE #3:GOTO 900
3600 ? #2;"Hit P to add points":? #2;"Hit L to add lines":? #2;"Hit Q to quit":GET #1,A:? #2,CHR$(A):RETURN
5000 IF CS=1 THEN GOSUB 850
5002 GOSUB 5600:IF A=81 THEN RETURN
5003 IF A=86 THEN 5100
5004 IF A=67 THEN 5300
5005 IF A=68 THEN 5500
5008 GOTO 5002
5100 GOSUB 5700:IF A<>83 THEN 5120
5111 ? #2;"The coordinates of the station point  are:";:X=X0:Y=Y0:Z=Z0:GOSUB 950:GOSUB 700
5113 ? #2;"(";X;",";Y;",";Z;") in rectangular coordinates":? #2;" or:(";A;",";B;",";C;") in polar coordinates":GOTO 900
5120 IF A<>68 THEN 5130
5121 ? #2;"You are looking at angles of:":X=TP:Y=PP:GOSUB 700
5123 ? #2;"    ",X;" degrees in the (x,y) plane":? #2;"and ";Y;" degrees in the (r,z) plane":IF P=0 THEN 900
5125 ? #2;"Towards the point at:":X=XP:Y=YP:Z=ZP:GOSUB 950:GOSUB 700
5127 ? #2;"(";X;",";Y;",";Z;") in rectangular coordinates":? #2;" or:(";A;",";B;",";C;") in polar coordinates":GOTO 900
5130 IF A<>80 THEN 5160
5133 GOSUB 5800:IF A<>65 THEN 5140
5134 FOR I=1 TO NP:GOSUB 5153:NEXT I:GOTO 900
5140 IF A<>71 THEN 5150
5141 ? #2;"What is the first point that you want to see";:INPUT J:IF J>NP OR J<0 OR J>INT(J) THEN 5141
5143 ? #2;"What is the last point that you want  to see";:INPUT K:IF K>NP OR K<J OR K>INT(K) THEN 5143
5145 FOR I=J TO K:GOSUB 5153:NEXT I:GOTO 900
5150 IF A<>79 THEN 5158
5151 ? #2;"What point do you want to see";:INPUT I:IF I>NP OR I<0 OR I>INT(I) THEN 5151
5153 OPEN #3,4,0,P$:GOSUB 710:GOSUB 950:GOSUB 700:? #2;"The coordinates of ";D$;I
5155 ? #2;"are:(";X;",";Y;",";Z;") in rectangular":? #2;" or:(";A;",";B;",";C;") in polar":POINT #3,EP,EN:CLOSE #3:GOTO 900
5158 IF A=77 THEN RETURN
5159 GOTO 5131
5160 IF A<>76 THEN 5195
5163 GOSUB 5800:IF A<>65 THEN 5170
5164 FOR I=1 TO NL:GOSUB 5183:NEXT I:GOTO 900
5170 IF A<>71 THEN 5180
5171 ? #2;"What is the first line that you want  to see";:INPUT J:IF J>NL OR J<0 OR J>INT(J) THEN 5171
5173 ? #2;"What is the last line that you want  to see";:INPUT K:IF K>NL OR K<J OR K>INT(K) THEN 5173
5174 IF K>NL OR K<J OR K>INT(K) THEN 5173
5175 FOR I=J TO K:GOSUB 5183:NEXT I:GOTO 900
5180 IF A<>79 THEN 5192
5181 ? #2;"What line do you want to see";:INPUT I:IF I>NL OR I<0 OR I>INT(I) THEN 5181
5183 OPEN #3,4,0,L$:GOSUB 720:X=A:Y=B:Z=C:GOSUB 950:GOSUB 700:? #2;"The coordinates of the start of line number ";I
```

```
5185 ? #2;"are:(";X;",";Y;",";Z;") in rectangular":? #2;" or:(";A;",";B;",";C;") in polar"
5187 GOSUB 720:X=D:Y=E:Z=F:GOSUB 950:GOSUB 700:? #2;"The coordinates of the end of line number ";I
5189 ? #2;"are:(";X;",";Y;",";Z;") in rectangular":? #2;" or:(";A;",";B;",";C;") in polar":POINT #3,EL,EN:CLOSE #3:GOTO 900
5192 IF A=77 THEN RETURN
5193 GOTO 5131
5194 GOTO 900
5195 IF A<>65 THEN 5200
5196 ? #2;"The angles of vision are as follows:":X=TV:Y=PV:GOSUB 700
5198 ? #2;"    ",X;" degrees in the (x,y) plane":? #2;"and ";Y;" degrees in the (x,z) plane":GOTO 900
5200 IF A=77 THEN RETURN
5299 GOTO 5100
5300 GOSUB 5700:IF A<>83 THEN 5320
5311 GOSUB 5111:A$="the station point":GOSUB 990:XO=X:YO=Y:ZO=Z
5312 IF P=1 AND XO<>XP THEN X=XP-XO:Y=YP-YO:Z=ZP-ZO:GOSUB 950:TP=8:PP=C
5313 GOSUB 750:GOTO 900
5320 IF A<>68 THEN 5330
5321 GOSUB 5121:GOSUB 920:GOSUB 750:GOTO 900
5330 IF A<>80 THEN 5360
5332 GOSUB 5800:IF A<>65 THEN 5340
5333 FOR I=1 TO NP:GOSUB 5342:NEXT I:GOTO 900
5340 IF A<>79 THEN 5350
5341 ? #2;"What point do you want to see";:INPUT I:IF I>NP OR I<1 OR I>INT(I) THEN 5341
5342 GOSUB 5153:OPEN #3,12,0,P$:A$=D$:A$(14)=STR$(I):GOSUB 990:GOSUB 730:POINT #3,EP,EN:CLOSE #3:GOTO 900
5350 IF A<>71 THEN 5358
5351 ? #2;"What is the first point that you want to see";:INPUT J:IF J>NP OR J<0 OR J>INT(J) THEN 5351
5353 ? #2;"What is the last point that you want  to see";:INPUT K:IF K>NP OR K<J OR K>INT(K) THEN 5353
5355 FOR I=J TO K:GOSUB 5342:NEXT I:GOTO 900
5358 IF A=77 THEN RETURN
5359 GOTO 5300
5360 IF A<>76 THEN 5390
5362 GOSUB 5800:IF A<>65 THEN 5370
5363 FOR I=1 TO NL:GOSUB 5372:NEXT I
5370 IF A<>79 THEN 5380
5371 ? #2;"What line do you want to see";:INPUT I:IF I>NL OR I<1 OR I>INT(I) THEN 5371
5373 GOSUB 5183:A$=B$:A$(19)=STR$(I):GOSUB 990:D1=X:D2=Y:D3=Z:A$=C$:A$(17)=STR$(I):GOSUB 990
5379 A=D1:B=D2:C=D3:D=X:E=Y:F=Z:OPEN #3,12,0,L$:GOSUB 740:POINT #3,EL,EN:CLOSE #3:GOTO 900
5380 IF A<>71 THEN 5388
5381 ? #2;"What is the first line that you want to see";:INPUT J:IF J>NL OR J<0 OR J>INT(J) THEN 5381
5383 ? #2;"What is the last line that you want  to see";:INPUT K:IF K>NL OR K<J OR K>INT(K) THEN 5383
5385 FOR I=J TO K:GOSUB 5372:NEXT I:GOTO 900
5390 IF A<>65 THEN 5398
5391 GOSUB 5196:GOSUB 910:GOSUB 750:GOTO 900
5398 IF A=77 THEN RETURN
5399 GOTO 5300
5500 IF NP>0 THEN ? #2;"Hit P for points"
5505 IF NL>0 THEN ? #2;"Hit L for lines"
5510 ? #2;"Hit M for the main menu":GET #1,A:? #2,CHR$(A):IF A<>80 THEN 5550
5522 GOSUB 5800:IF A<>65 THEN 5527
5523 ? #2;"Type Y to delete all points";:GET #1,A:? #2,CHR$(A):IF A<>89 THEN RETURN
5525 NP=0:GOTO 900
5527 IF A<>71 THEN 5540
5528 ? #2;"What point do you want to start with";:INPUT J:IF J>INT(J) OR J<1 OR J>NP THEN 5528
5530 ? #2;"What point do you want to end with";:INPUT K:IF K>INT(K) OR K<J OR K>NP THEN 5530
5532 ? #2;"Type Y to delete points ";J;" through ";K;:GET #1,A:? #2,CHR$(A):IF A<>89 THEN RETURN
```

```
5535 OPEN #3,12,0,P$:FOR R=@0 TO NP-K:I=K+R:GOSUB 710:I=J+R:GOSUB 730:NEXT R:POINT #3,EP,EN:CLOSE #3:NP=NP-K+J:GOTO 900
5540 IF A<>79 THEN 5548
5541 ? #2;"What point do you want to delete";:INPUT A:IF A>INT(A) OR A<1 OR A>NP THEN 5541
5544 ? #2;"Type Y to delete point ";A:GET #1,B:? #2,CHR$(B):IF B<>89 THEN RETURN
5545 OPEN #3,12,0,P$:FOR R=A TO NP-@9:I=R+@9:GOSUB 710:I=R:GOSUB 730:NEXT R:POINT #3,EP,EN:CLOSE #3:NP=NP-@9:GOTO 900
5548 IF A=77 THEN RETURN
5549 GOTO 900
5550 IF A<>76 THEN 5580
5552 GOSUB 5800:IF A<>65 THEN 5557
5554 ? #2;"Type Y to delete all lines":GET #1,A:? #2,CHR$(A):IF A<>89 THEN RETURN
5556 NL=0:GOTO 900
5557 IF A<>71 THEN 5570
5558 ? #2;"What line do you want to start with";:INPUT J:IF J>INT(J) OR J<1 OR J>NP THEN 5558
5560 ? #2;"What line do you want to end with";:INPUT K:IF K>INT(K) OR K<J OR K>NP THEN 5560
5562 ? #2;"Type Y to delete lines ";J;" through ";K;:GET #1,A:? #2,CHR$(A):IF A<>89 THEN RETURN
5565 OPEN #3,12,0,L$:FOR R=@0 TO NL-K:I=K+R:GOSUB 720:I=J+R:GOSUB 740:NEXT R:POINT #3,EL,EN:CLOSE #3:NL=NL-K+J:GOTO 900
5570 IF A<>79 THEN 5578
5571 ? #2;"What line do you want to delete";:INPUT A:IF A>INT(A) OR A<1 OR A>NP THEN 5571
5573 ? #2;"Type Y to delete point ";A:GET #1,B:? #2,CHR$(B):IF B<>89 THEN RETURN
5575 OPEN #3,12,0,L$:FOR R=A TO NL-@9:I=R+@9:GOSUB 720:I=R:GOSUB 740:NEXT R:POINT #3,EL,EN:CLOSE #3:NL=NL-@9:GOTO 900
5578 IF A=77 THEN RETURN
5579 GOTO 900
5580 IF A=77 THEN RETURN
5585 GOTO 5500
5599 RETURN
5600 ? #2;"Hit V to view coordinates":? #2;"Hit C to change coordinates":? #2;"Hit D to delete coordinates"
5610 ? #2;"Hit Q to quit":GET #1,A:? #2,CHR$(A):RETURN
5700 ? #2;"Hit S for the station point":? #2;"Hit D for direction of viewing":IF NP>0 THEN ? #2;"Hit P for points"
5720 IF NL>0 THEN ? #2;"Hit L for lines"
5725 ? #2;"Hit A for angles of vision":? #2;"Hit M for the main menu":GET #1,A:? #2,CHR$(A):RETURN
5800 ? #2;"Hit A for all":? #2;"Hit O for one":? #2;"Hit G for a group":? #2;"Hit M for the main menu"
5825 GET #1,A:? #2,CHR$(A):RETURN
8000 NP=0:NL=0:FN=1:? #2;"}Type 1 for rectangular":? #2;"  or 2 for polar coordinates for the":? #2;"function";:INPUT FC
8001 CS=0:IF FC<1 OR FC>2 OR FC>INT(FC) THEN 8000
8002 A$="the station point ":GOSUB 990:X0=X:Y0=Y:Z0=Z:GOSUB 920:GOSUB 910:GOSUB 750
8022 GOSUB 8900:? #2;"What is the initial x value";:INPUT FX:? #2;"What size are the x increments";:INPUT DX
8024 ? #2;"How many x increments are to be used";:INPUT NX:IF NX<0 OR NX>INT(NX) THEN 8024
8027 ? #2;"What is the initial y value";:INPUT FY:? #2;"What size are the y increments";:INPUT DY
8031 ? #2;"How many y increments are to be used";:INPUT NY:IF NY<0 OR NY>INT(NY) THEN 8031
8040 ? #2;"Type in the function in the form":? #2;"z=f(x,y).  Since the computer has to"
8042 ? #2;"read this, please use only capital":? #2;"letters.  Use * to multiply and / to"
8044 ? #2;"divide.  If your computer has THE":? #2;"FAST CHIP use SIN(x)/COS(x) instead"
8046 ? #2;"of TAN(x).  PI=3.14159265 but angles":? #2;"are measured in degrees.  What is":? #2;"your function";
8055 INPUT A$:POKE 559,0:? "}":POSITION 2,4:? 8820;A$:? "CONT":POSITION 2,0:POKE 842,13:STOP
8078 POKE 842,12:? "}";:GOSUB 8800:POKE 559,34:RETURN
8800 OPEN #3,12,0,P$:I=0:TRAP 8850:FOR R=@0 TO NX:FOR S=@0 TO NY:OI=I:I=R*NY+R+S+@9:X=FX+R*DX:Y=FY+S*DY:GOSUB 8820
8810 GOSUB 780:NEXT S:NEXT R:TRAP 400:GOSUB 790:GOSUB 750:RETURN
8825 IF FC=2 THEN A=Z:B=X:C=Y:GOSUB 960
8830 RETURN
8850 TRAP 400:IF PEEK(195)<>11 THEN 400
8855 TRAP 8890
8860 J=DX/@4:K=DY/@4:X=X-J:Y=Y-K:GOSUB 8820:D1=Z:Y=Y+K+K:GOSUB 8820:D2=Z:X=X+J+J:GOSUB 8820:D3=Z
8870 Y=Y-K-K:GOSUB 8820:D4=Z:X=X-J:Y=Y+K:Z=(D1+D2+D3+D4)/4:TRAP 8850:RETURN
8890 ? #2;"Your function won't work":GOTO 900
```

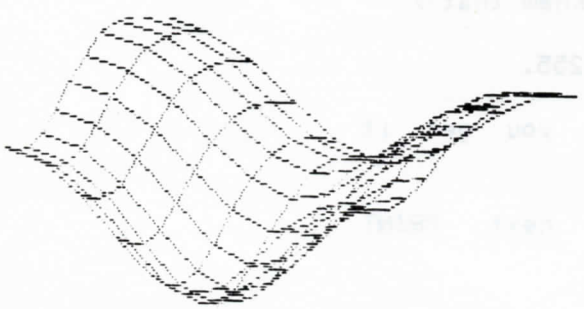
```
8900 IF FC=2 THEN ? #2;"From here on in we will refer to thetaas x, phi as y and the radius as z   [z=f(x,y)]"
8915 ? #2;"Type 0 to just plot points":? #2;"     1 to connect points with same x"
8917 ? #2;"     2 to connect points with same y":? #2;"     3 to connect points in a grid";
8920 INPUT FT:IF FT<0 OR FT>3 THEN 8915
8925 RETURN
```

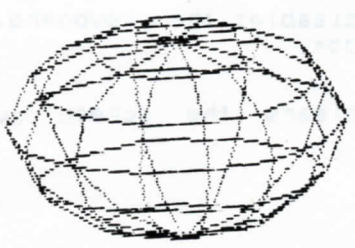
APPENDIX B - VARIABLE LIST

Below is a list of every variable used by my program. Each variable is listed in the same order as it appears on the variable table used by BASIC to keep track of the variables in the program. The name of each variable is given, followed by its uses in the program. The list is as follows:

MEM: Since this is the first variable in the program, it is used to get the 6-byte floating point representation used by the disk files.
A$: This string is used as a multipurpose string variable.
B$: This is used to store the string "the start of line ".
C$: This is used to store the string "the end of line ".
D$: This is used to store the string "point number ".
P$: This is used to store the string "D:POINTS".
L$: This is used to store the string "D:LINES".
S(n): This used to store the values used in I/O with the disk files.
SM: This stores to address of the start of the variable value table.
MP: This stores the maximum number of points for the program (2430).
ML: This stores the maximum number of lines for the program (1243).
SP: This stores the starting disk sector of D:POINTS (369).
SL: This stores the starting disk sector of D:LINES (4).
Q: This stores the often used number 256.
Q0: This stores the often used number 0.
Q3: This stores the often used number 0.5 (1/2).
Q4: This stores the often used number 100.
Q9: This stores the often used number 1.
CX: This stores the x coordinate for the center of the graphics screen (159).
CY: This stores the y coordinate for the center of the graphics screen (95).
GC: This stores the correction factor for plotting due to the fact that graphics pixels are rectangular (0.64706).
EP: This stores the ending disk sector for D:POINTS (718).
EL: This stores the ending disk sector for D:LINES (359).
PI: This stores the value of the mathematical constant pi (3.14159265).
A: This is used as a common numerical variable and as the radius part of a polar (spherical) coordinate.
I: Used as a counting variable.
X: Used as a common variable and as a x coordinate for a cartesian point.
Y: Used as a common variable and as a y coordinate for a cartesian point.
Z: Used as a common variable and as a z coordinate for a cartesian point.
B: Used as a common variable and as the x,y angle (theta) for a polar (spherical) point.
C: Used as a common variable and as the (xy),z angle (phi) for a polar (spherical) point.
D: Used to store the x coordinate of the end of a line.
E: Used to store the y coordinate of the end of a line.
F: Used to store the z coordinate of the end of a line.
J: Used as a counting and common variable.

K: Used as a counting and common variable.
XO: Used to store the x coordinate for the station point.
YO: Used to store the y coordinate for the station point.
ZO: Used to store the z coordinate for the station point.
TP: Used to store the horizontal direction of viewing.
PP: Used to store the vertical direction of viewing.
XP: Used to store the x coordinate for the point that you are looking towards.
YP: Used to store the y coordinate for the point that you are looking towards.
ZP: Used to store the z coordinate for the point that you are looking towards.
TV: Used to store the horizontal angle for the field of view.
PV: Used to store the vertical angle for the field of view.
XS: Used to store the x offset for the station point (see app.D).
FX: Used to store the initial x value for a function.
DX: Used to store the size of x increment for a function.
NX: Used to store the number of x increments for a function.
FY: Used to store the initial y value for a function.
DY: Used to store the size of y increment for a function.
NY: Used to store the number of y increments for a function.
NP: Used to store the number of points in the non-functional system.
NL: Used to store the number of lines in the non-functional system.
CS: Flag-0=Original system, 1=Changed system
FN: Flag-0=Non-function, 1=Fuction
FT: Used to indicate how to plot a functional system.
OI: Used to store the previous value of I.
SY: The scale for y coordinates determined by TV&PV (see app.D).
P: Flag-0=Looking in a direction, 1=Looking towards a point
D3: Used to temporarily store numbers.
D4: Used to temporarily store numbers.
D1: Used to temporarily store numbers.
D2: Used to temporarily store numbers.
N: Used as a common variable.
G: Used as a common variable.
H: Used as a common variable.
R: Used as a common variable and a counting variable.
S: Used as a common variable and a counting variable.
FC: FLAG-1=Cartesian function,2=Spherical function





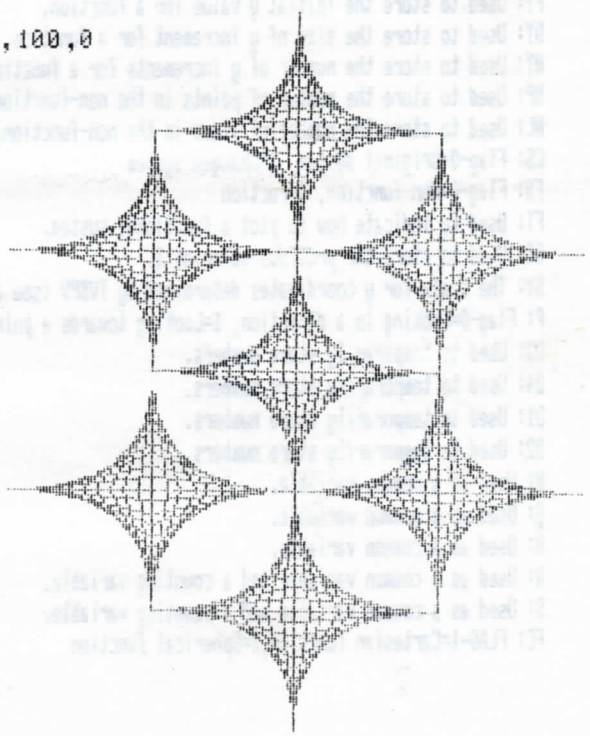
STARS

PACUS MAY 1984

The short program presented here was adapted from an Apple/TRS-80 program printed in an article titled "Plotter Tutorial Part 3 - Thinking In 3-D" which appeared in the February 1984 issue of Creative Computing. Although this program does not produce a 3-D figure (it was the first listing in the article) it does illustrate the technique of recursive use of a simple pattern to create very interesting designs.

Geo Ruiter

```
100 DIM A$(10)
110 GRAPHICS 24:XC=159:YC=96:XM=0.9:YM=0.9
120 SETCOLOR 2,2,6:SETCOLOR 4,2,6:SETCOLOR 1,2,0:COLOR 1
130 FOR N=1 TO 7:READ X,Y
140 CX=X:CY=Y:GOSUB 190
150 NEXT N
160 GOTO 160
170 STOP
180 DATA 0,0,-50,-50,50,-50,-50,50,50,50,-100,0,100,0
190 FOR C=1 TO 4
200 FOR X2=0 TO 50 STEP 5
210 Y=50-X2:X1=X2
220 IF C=2 THEN Y=-Y
230 IF C=3 THEN Y=-Y:X1=-X2
240 IF C=4 THEN X1=-X2
250 A$="MOVE":X=CX:Y=CY+Y:GOSUB 290
260 A$="DRAW":X=CX+X1:Y=CY:GOSUB 290
270 NEXT X2
280 NEXT C:RETURN
290 XP=X*XM+XC:YP=192-(Y*YM+YC)
300 IF A$="MOVE" THEN XO=XP:YO=YP:RETURN
310 PLOT XO,YO:DRAWTO XP,YP
320 XO=XP:YO=YP:RETURN
```



Tid-Bits

POKE 752,1 turns the cursor off (Betcha EVERYBODY knew that!)

PEEK 53770 returns a random integer between 0 and 255.

POKE 16,255 disables the keyboard. (But how do you get it "re-enabled???")

POKE 87,1 clears the screen just before the next PRINT statement.

421-2159

NEWSLETTER INFORMATION:

This newsletter is written and printed by members of the Milwaukee Area ATARI User's Group (MILATARI), an association of individuals with a common interest in using and programming ATARI computers. MILATARI is not affiliated with the ATARI company or any other commercial organizations.

All articles are written and donated by the membership. Opinions expressed in this publication are those of the individual author and do not necessarily represent, nor reflect, the opinions of MILATARI nor those of any other commercial or non-commercial organizations. Any article appearing in this newsletter may be reproduced, providing credit is given to the author and to MILATARI.

Your contribution of articles are always welcome. You may submit your article on ATARI compatible cassette, diskette, on typewritten form, or you can arrange with the editor to download your file via a modem at either 300 or 1200 BAUD. When submitting your article on cassette or diskette, please do not include any format control codes imbedded within the text. Deadline for articles is the last day of each month for inclusion on the next issue.

Write MILATARI NEWSLETTER, P.O. Box 1191, Waukesha, WI 53187-1191 for more information.

MEMBERSHIP INFORMATION:

Membership is open to individuals and families who are interested in using and programming ATARI computers. The membership includes a subscription to this newsletter and access to the clubs cassette, diskette and publication libraries.

There are 3 classes of memberships available. Associate, Individual and Family. Associate members can attend all club functions and may withdrawal materials from the club libraries. In addition to attending club functions and checking out materials from the libraries, Individual and Family members are entitled to vote in club elections and to hold elected position in the organization. The annual membership fees are $10.00 for associate, $15.00 for individual, and $20.00 for the family membership. Members are expect to abide by the by-laws of the club. You may receive a copy of the by-laws by contacting the club secretary.

For more information on how to join MILATARI, please contact the membership committee.

MEETING INFORMATION:

MILATARI meetings are held once monthly. The meetings are currently being held at the Armbruster School, 7000 Greenway, Greenfield. (Off 68th Street, behind Southridge Shopping Center.) The date of the meeting is the third Saturday of each month. Doors are open at 2:00PM. An agenda of the next meeting can be found elsewhere in this newsletter. For more specific details on the agenda for the next scheduled meeting, please contact the Vice President.

MILATARI Officers:

President	Gary Nolan	353-9716
Vice President	Chris Stieber	529-2663
Treasurer	David Frazer	542-7242
Secretary	open	
Education Committee	Linda Scott (Chairperson)	466-2314
	Joe Sanders (SIG Chairman)	447-1660
	Erick Hanson (Instructor)	252-3146
Cassette Librarian	Ron Friedel	354-1717
Disk Librarian	Bill Lawrence	968-3082
	Carl Mielcarek	355-3539
Publication Librarian	Marcus Hagen	445-0710
Membership Committee	Dennis J. Bogie	968-9341
	Sharon Gamache	421-2887
Newsletter Editor	David Frazer	542-7242
Bulletin Board SYSOP	Pete Kurth	355-6031(BBS)

****

TECHNICAL SUPPORT GROUP:

The following members have indicated a willingness to assist MILATARI members with programming and other related technical problems. Please be polite and do not call these members during meal periods or at very early or very late hours.

William Lawrence	Programming	1-968-3082
Don Wilcox	Programming	228-1650
Erik Hanson	Prog/Tech	252-3146
Steve Booth	Programming	367-8739
Nick Liberski	Prog/Tech	782-5594
David Frazer	Prog/Tech	542-7242

****

MILATARI BULLETIN BOARD:

The Milwaukee Area ATARI Users Group maintains a 24 hour bulletin board service. This board is designed for the use of our members and other ATARI users around the country. The BBS allows for upload and downloading programs and files, a public message board and club news. The board operates at 300 BAUD. The phone number is (414)355-6031.

==========

MILATARI NEWSLETTER
P.O. Box 1191
Waukesha, Wisconsin 53187-1191

==========

ADDRESS CORRECTION REQUESTED

FORWARDING POSTAGE GUARANTEED

==========